《C/C++学习指南》

第12.2讲：malloc/free用法示例

作者：邵发　　QQ群：417024631
官网：http://www.afanihao.cn/c_guide/
答疑：http://www.afanihao.cn/kbase/

何为“对象”

对象指的一块内存

```
Contact a;  // a是一个对象，即存放着一个对象的数据
Contact* p = (Contact*) malloc(sizeof(Contact));
```

确切地说：p指向了一个对象

malloc/free示例

示例：用Citizen表示一个市民，用Car表示一个辆车。他起初没有车，但未来可能有一辆车。

```
struct Car
{
    char maker[32]; // 制造商
    int  price;  // 价格
};
struct Citizen
{
    char name[32]; // 名字
    int  deposite; // 存款
    Car* car;  // NULL时表示没车
};
```


malloc/free示例

定义一个对象,开始没车

```
Citizen shaofa = { “shaofa”, 100, NULL };
```

后来，他可能买了一辆车

```
void buy(Citizen* owner)
{
    // 创建一个对象
    Car* car = (Car*) malloc(sizeof(Car));
    strcpy(car->maker, “Chevrolet”);
    car->price = 10;

    // 保存此对象（确切地说是记住了指针）
    owner->car = car; // 有车了
    owner->deposite -= car->price; // 钱没了
}
```

malloc/free示例

终有一天，这车会报废。。。

```
void discard(Citizen* who)
{
    free(who->car); // 此对象被销毁
    who->car = NULL;  // 回到无车状态
}
```


malloc/free示例

也有可能会买给别人，

```
void sell(Citizen* owner, Citizen* other)
{
    Car* car = owner->car;
    car->price *= 0.5; //半价出售
    other->car = car; // 别人拥有了这辆车

    owner->deposite += car->price; // 收回一部分成本
    //free(car); // oh,no! 不能free，这车在别人手里
    owner->car = NULL;  // 回到无车状态
}
```

注意事项

(1)不是malloc的指针，不可以free

例如，

```
int a = 10;
int* p = a;
free (p ); // 开什么玩笑？？这个指针根本不是从MM那里借来的
```


注意事项

（2） malloc的内存，必须及时free

怎么样才算“及时”？

“不及时”会怎样？

MM里可用的内存是有限的，你用完了就得尽快还，因为别的应用程序也需要MM的内存。

只借不还，积累到一定程度，MM没有更多内存可用，于是malloc返回NULL。

```
while(1)
{
   void* ptr = malloc(1024*512);
}
```

注意事项

（3） 要free必须free首地址

错误的例子：

```
char* p = (char*) malloc(100); // 100个字节
free (p+50); // 借了100, 只还50?
```

要还就得全还，否则MM那边处理不了。(p, n)

注意事项

（4） malloc的返回值需要检测

```
char* ptr = (char*) malloc(1024); // 512K
if(ptr != NULL)
{
...
}
```

原因是：MM可能此时没有闲置内存可用。（虽然这种情况一般不会发生）

注意事项

（5） free之后，该指针不应再使用
free之后，该内存交还给MM，该内存不再可用（失效）。
以下代码是错误的：

```
char* p = (char*) malloc(100);
free (p);
for(int i=0; i<100; i++)
{
   p[i] = i; //该内存已经被free，绝不可继续使用！
}
```

良好的编程习惯是：

```
free(p);
p = NULL;  // 置为空指针
```


注意事项

（6） malloc得到的内存，可以任意位置释放
不一定要在相同的函数里释放，在应用程序的任意一个角落释放都是有效的。

也就是说：这一块内存被malloc出来之后，完全交给你处置。